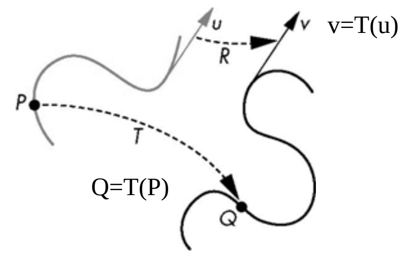
Transformations

Objectives

- Introduce standard transformations
  - Rotation
  - Translation
  - Scaling
  - Shear
- Derive homogeneous coordinate transformation matrices
- Learn to build arbitrary transformation matrices from simple transformations

General Transformations

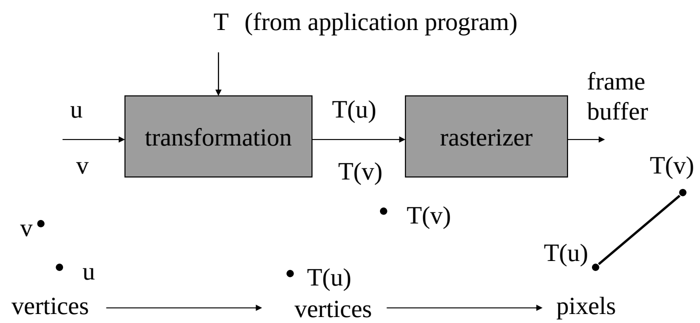
A transformation maps points to other points and/or vectors to other vectors



Affine Transformations

- Line preserving
- Characteristic of many physically important transformations
  - Rigid body transformations: rotation, translation
  - Scaling, shear
- Importance in graphics is that we need only transform endpoints of line segments and let implementation draw line segment between the transformed endpoints

Pipeline Implementation



Notation

We will be working with both coordinate-free representations of transformations and representations within a particular frame

P,Q, R: points in an affine space

u, v, w: vectors in an affine space

α, β, γ: scalars

p, **q**, **r**: representations of points

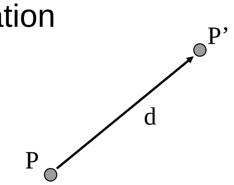
-array of 4 scalars in homogeneous coordinates

u, **v**, **w**: representations of points

-array of 4 scalars in homogeneous coordinates

Translation

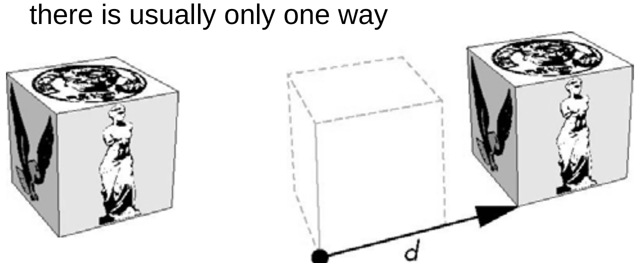
- Move (translate, displace) a point to a new location



- Displacement determined by a vector d
  - Three degrees of freedom
  - P'=P+d

How many ways?

Although we can move a point to a new location in infinite ways, when we move many points there is usually only one way



object

translation: every point displaced by same vector

Translation Using Representations

Using the homogeneous coordinate representation in some frame

$\mathbf{p}=[\ x\ y\ z\ 1]^T$

$\mathbf{p}'=[x'\ y'\ z'\ 1]^T$

$\mathbf{d}=[dx\ dy\ dz\ 0]^T$

Hence $\mathbf{p}' = \mathbf{p} + \mathbf{d}$ or

$x'=x+d_x$

$y'=y+d_y$

$z'=z+d_z$

note that this expression is in four dimensions and expresses point = vector + point

Translation Matrix

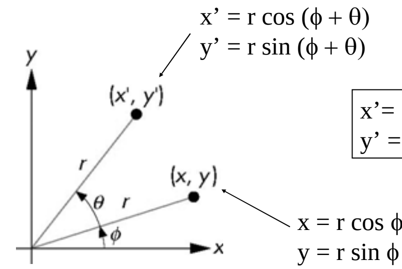
We can also express translation using a
4 x 4 matrix **T** in homogeneous coordinates
p'=**Tp** where

$$\mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This form is better for implementation because all affine transformations can be expressed this way and multiple transformations can be concatenated together

Rotation (2D)

Consider rotation about the origin by θ degrees
- radius stays the same, angle increases by θ



$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Rotation about the z axis

- Rotation about z axis in three dimensions leaves all points with the same z
  - Equivalent to rotation in two dimensions in planes of constant z

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

  - or in homogeneous coordinates

$$\mathbf{p}' = \mathbf{R}_z(\theta)\mathbf{p}$$

Rotation Matrix

$$\mathbf{R} = \mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about x and y axes

- Same argument as for rotation about *z* axis
  - For rotation about *x* axis, *x* is unchanged
  - For rotation about *y* axis, *y* is unchanged

$$\mathbf{R} = \mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R} = \mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

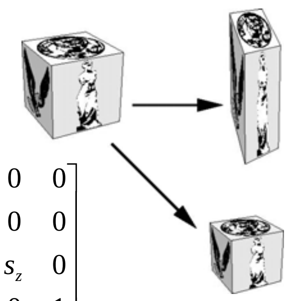
Scaling

Expand or contract along each axis (fixed point of origin)

$x' = s_x x$
$y' = s_y x$
$z' = s_z x$

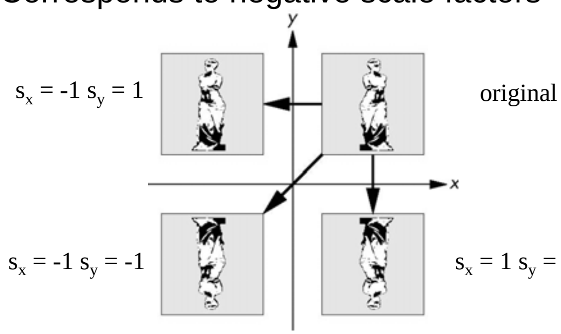
$\mathbf{p}' = \mathbf{S}\mathbf{p}$

$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Reflection

Corresponds to negative scale factors



Inverses

- Although we could compute inverse matrices by general formulas, we can use simple geometric observations
  - Translation: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathbf{T}(-d_x, -d_y, -d_z)$
  - Rotation: $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$
    - Holds for any rotation matrix
    - Note that since $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$

    $\mathbf{R}^{-1}(\theta) = \mathbf{R}^{T}(\theta)$
  - Scaling: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$

Concatenation

- We can form arbitrary affine transformation matrices by multiplying together rotation, translation, and scaling matrices
- Because the same transformation is applied to many vertices, the cost of forming a matrix **M**=**ABCD** is not significant compared to the cost of computing **Mp** for many vertices **p**
- The difficult part is how to form a desired transformation from the specifications in the application

Order of Transformations

- Note that matrix on the right is the first applied
- Mathematically, the following are equivalent

$$\mathbf{p}' = \mathbf{ABCp} = \mathbf{A}(\mathbf{B}(\mathbf{Cp}))$$

- Note many references use column matrices to represent points. In terms of column matrices

$$\mathbf{p}'^{T} = \mathbf{p}^{T}\mathbf{C}^{T}\mathbf{B}^{T}\mathbf{A}^{T}$$

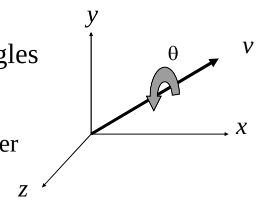
General Rotation About the Origin

A rotation by θ about an arbitrary axis can be decomposed into the concatenation of rotations about the *x*, *y*, and *z* axes

$$\mathbf{R}(\theta) = \mathbf{R}_z(\theta_z)\ \mathbf{R}_y(\theta_y)\ \mathbf{R}_x(\theta_x)$$

$\theta_x\ \theta_y\ \theta_z$ are called the Euler angles

Note that rotations do not commute
We can use rotations in another order
but with different angles



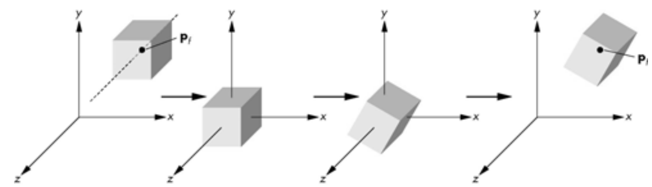
Rotation About a Fixed Point other than the Origin

Move fixed point to origin

Rotate

Move fixed point back

$\mathbf{M} = \mathbf{T}(p_f)\ \mathbf{R}(\theta)\ \mathbf{T}(-p_f)$



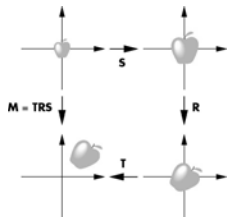
Instancing

- In modeling, we often start with a simple object centered at the origin, oriented with the axis, and at a standard size
- We apply an *instance transformation* to its vertices to

    Scale

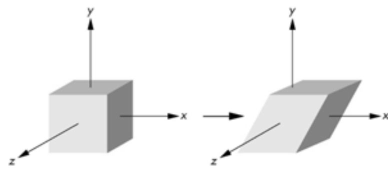
    Orient

    Locate



Shear

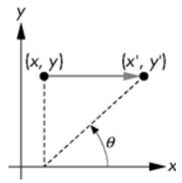
- Helpful to add one more basic transformation
- Equivalent to pulling faces in opposite directions



Shear Matrix

Consider simple shear along *x* axis

$x' = x + y \cot \theta$
$y' = y$
$z' = z$



$$\mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$H^{-1}(\theta) = H(-\theta)$